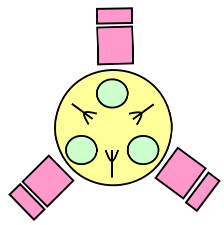
Σειρά Προβλημάτων 1

Ημερομηνία Παράδοσης: 3/10/17

<u>Άσκηση 1</u>

N φιλόσοφοι βρίσκονται σε μία αίθουσα και περνούν τον περισσότερό τους χρόνο σε διαλογισμό φιλοσοφικών θεωριών. Στη διπλανή αίθουσα βρίσκεται ένα τραπέζι με *N* καρέκλες, *N* πιάτα και *N* πιρούνια. Έτσι, όταν κάποιος φιλόσοφος πεινάσει κάθεται στην καρέκλα του, παίρνει τα δύο πιρούνια που βρίσκονται δίπλα από το πιάτο του, και τρώει. Όταν τελειώσει αφήνει τα δύο πιρούνια και επιστρέφει στην αίθουσα διαλογισμού.



Προφανώς είναι αδύνατο να τρώνε ταυτόχρονα όλοι οι φιλόσοφοι όπως επίσης είναι αδύνατο να τρώνε ταυτόχρονα δύο φιλόσοφοι που κάθονται σε γειτονικές θέσεις.

Το πιο κάτω πρωτόκολλο προτείνει μια τεχνική που μπορεί να χρησιμοποιηθεί από τους φιλόσοφους κατά τη διάρκεια των γευμάτων τους: επανειλημμένα κάθε ένας από αυτούς σηκώνει αρχικά το πιρούνι που βρίσκεται στα αριστερά του και στη συνέχεια αυτό που βρίσκεται στα δεξιά του. Στον πιο κάτω κώδικα υποθέτουμε ότι τα πιρούνια μοντελοποιούνται από τον πίνακα

```
bool fork[n]:= [false,..,false];
```

όπου `fork[i]` είναι το i-οστό πιρούνι και ισχύει `fork[i] = false` αν το πιρούνι είναι διαθέσιμο και `fork[i] = true` αν το πιρούνι βρίσκεται σε χρήση. Ο φιλόσοφος i, όπου $0 \leq i < n$, κωδικοποιείται ως εξής:

<u>`P[i]`</u>

```
1. while true do{
2.       atomic{ if (fork[i] = false
3.                  then fork[i] = true;
4.                else
5.                  goto line 2}
6.       atomic{ if (fork[(i+1)mod n] = false
7.                  then fork[(i+1)mod n] = true;
8.                else
9.                  goto line 6}
10. //eat;
11. fork[i] = false;
12. fork[(i+1)mod n] = false;
13. }
```

(α) Μοντελοποιήστε το πρωτόκολλο με δύο φιλόσοφους (δηλαδή, για n=2) ως ένα γράφο προγράμματος.

(β) Αποδώστε το σύστημα μεταβάσεων του πρωτοκόλλου γραφικά.

(γ) Συζητήστε την καταλληλότητα του πρωτοκόλλου για την ομαλή διεξαγωγή των γευμάτων των φιλοσόφων.

Άσκηση 2

Θεωρήστε τις δύο πιο κάτω παράλληλες διεργασίες:

Διεργασία 1

```
0:  if lock = 0  then lock:= 1;
1:  x := 1;
2:  if lock = 1 then lock:= 0;
3:
```

Διεργασία 2

```
0: if lock = 0 then lock:= 1;
1: x := 2;
2: if lock = 1 then lock:= 0;
3:
```

(α) Μοντελοποιήστε τη σύνθεση των δύο διεργασιών ως ένα γράφο προγράμματος και αποδώστε τον γραφικά ως ένα σύστημα μεταβάσεων.

(β) Αποφασίστε κατά πόσο το σύστημα ικανοποιεί τις πιο κάτω ιδιότητες, όπου pc_1 και pc_2 οι program counter των διεργασιών 1 και 2 αντίστοιχα.

(i) $F \langle pc_1 = 3\rangle$

(ii) $G (\langle lock = 1\rangle \rightarrow F \langle lock = 0\rangle)$

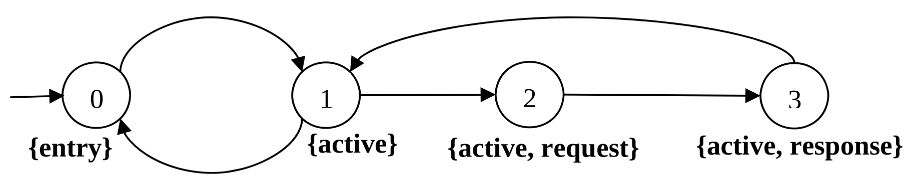
(iii) $G (\langle pc_1 = 2\rangle \rightarrow X \langle pc_1 = 3\rangle)$

(iv) $F (\langle pc_1 = 1\rangle \wedge \langle pc_2 = 1\rangle)$

(v) $G (\langle pc_1 = 3\rangle \rightarrow G \langle pc_1 = 3\rangle)$

Άσκηση 3

Θεωρήστε την ακόλουθη δομή Kripke.



Να διατυπώσετε τις πιο κάτω προτάσεις στην LTL (αν αυτό είναι εφικτό) και να αποφασίσετε κατά πόσο ικανοποιούνται στην αρχική κατάσταση της δομής.
(α) Η ατομική πρόταση active ικανοποιείται απείρως συχνά.
(β) Η ατομική πρόταση entry ικανοποιείται απείρως συχνά.
(γ) Είναι πάντα δυνατή η ικανοποίηση της ατομικής πρότασης entry.
(δ) Κάθε αίτημα (request) ακολουθείται από μια ανταπόκριση (response).
(ε) Αν από κάποια στιγμή και μετά δεν υπάρξει καινούριο request, τότε θα ικανοποιηθεί απείρως συχνά η ατομική πρόταση entry.
(ζ) Η ατομική πρόταση response ικανοποιείται μόνο εφόσον έχει προηγηθεί ικανοποίηση της ατομικής πρότασης request.

Άσκηση 4

Να ελέγξετε ποιες από τις πιο κάτω ιδιότητες αποτελούν ταυτολογίες χρησιμοποιώντας τη σημασιολογία της LTL δίνοντας είτε απόδειξη της συνεπαγωγής είτε κάποιο αντιπαράδειγμα δομής Kripke στην οποία να ικανοποιείται η μία ιδιότητα αλλά όχι η άλλη.

i. $\mathbf{X} (a \vee \mathbf{F}\, a) \rightarrow \mathbf{F}\, a$

ii. $\mathbf{F}\, a \rightarrow \mathbf{X} (a \vee \mathbf{F}\, a)$

iii. $\mathbf{GF}\, p \wedge \mathbf{FG}\, q \rightarrow \mathbf{FG} (\mathbf{F}\, p \vee q)$

iv. $(\mathbf{GF}\, p \vee \mathbf{FG}\, q) \rightarrow \mathbf{FG} (\mathbf{F}\, p \wedge q)$

Άσκηση 5

Θεωρήστε ένα σύνολο από ενέργειες Act και, για κάθε $\alpha \in$Act, τις ατομικές προτάσεις *enabled_α* και *executed_α*, όπου η πρόταση *enabled_α* ικανοποιείται σε μία κατάσταση αν από τη συγκεκριμένη κατάσταση η ενέργεια *α* είναι εκτελέσιμη και η πρόταση *executed_α* ικανοποιείται σε μία κατάσταση αν η αμέσως προηγούμενη ενέργεια που έτυχε εκτέλεσης ήταν η *α*.

Ένα μονοπάτι ονομάζεται *ασθενώς δίκαιο σε σχέση με την ενέργεια α* αν δεν υπάρχει σημείο στο μονοπάτι από το οποίο και μετά η *α* είναι συνεχώς εκτελέσιμη αλλά δεν εκτελείται ποτέ.

Ένα μονοπάτι ονομάζεται *ισχυρά δίκαιο σε σχέση με την ενέργεια α* αν δεν ισχύει ότι υπάρχει σημείο στο μονοπάτι από το οποίο και μετά ενώ η *α* είναι εκτελέσιμη απείρως συχνά εκτελείται μόνο για πεπερασμένο αριθμό φορών.

Έστω φ ιδιότητα της LTL και ενέργεια $\alpha \in$Act. Να ορίσετε δύο ιδιότητες της LTL οι οποίες να εκφράζουν (i) ότι η φ ικανοποιείται σε όλα μονοπάτια που είναι ασθενώς δίκαια σε σχέση με την *α* και (ii) ότι η φ ικανοποιείται σε όλα μονοπάτια που είναι ισχυρά δίκαια σε σχέση με την *α*.